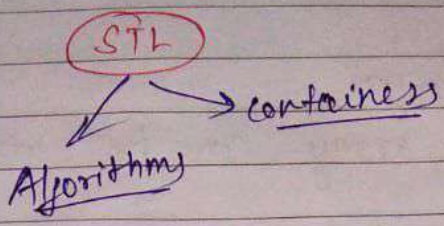
Lec-09

*. C++ STL



STL Array

```
#include <iostream>
using namespace std;
int main(){
    array<int,4> a = {1,2,3,4};
    for(int i=0 ; i<a.size; i++){
        cout << a[i] << " ";
    }
    cout << endl;
cout << "element at 2nd index -> " << a.at(2) << endl;
cout << " Empty or not -> " << a.empty() << endl;
cout << " First element" << a.front() << endl;
cout << "Last element -> " << a.back() << endl;
```

o/p screen.

```
1 2 3 4
Element at 2nd index -> 3
Empty or not -> 0
First element 1
last element -> 4
```

→ STL array is a static array, thats why not frequently used.

Vector

- Dynamic array.
  i.e. size of array can be increased or decreased.
- vector <int> v;
- vector <int> v(5);
  (5 → size)
- #include <vector>
  To include vector.

- v.size() // How many element is present.
- v.capacity() // How many elements can be stored.
- v.push_back(2);
- v.pop_back(2);
- arr[] → | 1 | 2 | 3 | 4 | 5 |
  (0 1 2 3 4)

vector <int> arr;
arr.front() → O/P → 1
arr.back() → O/P → 5

- v.clear() :- Empty the vector, but remember capacity remains same as before, although size will be 0.

- vector <int> v;
  vector <int> a(v); // copy all the element of v in a.

- vector <int> v(5, 2);
  | 2 | 2 | 2 | 2 | 2 |
  (0 1 2 3 4)
  All elements will be initialized by 2.

```
for(int i : v){
    cout << i << " ";
}
```

O/P screen
2 2 2 2 2

- Everytime when vector gets full and we need to add more elements, vector get double its size.
- v.at(i); // To access element at a particular index.

→ Deque

- #include <deque>
  To include deque.
- deque <int> d;
- d.clear() // clear all
- d.push_back(1); // 1
- d.push_front(2); // 21
- d.push_back(4); // 214
- d.pop_front(); // 14
- d.pop_back(); // 1
- d.erase(d.begin(), d.begin()+1);
  In erase we need to tell that which element we want to erase.

→ List

- #include <list>
  ⋮
  list<int> l;
- l.push_back(1);
  l.push_front(2);
- Can perform all functions we discussed in vector and deque.

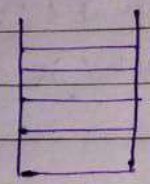
- Cannot access an element at a particular index directly.
  We have to traverse to access that element.
  i.e. l.at(i) ✗
  We cannot use it.

→ Stack

- #include <stack>

```
stack <string> s;
```

- s.push("Harsh");
  s.push("kunal");
- s.pop()
- s.top();
  Print top element in the stack.
- s.size(); // size of stack
- s.empty(); // Empty or not.



→ Stack follows LIFO structure
↳ Last In First Out.

→ Queue

- Queue follows FIFO structure.
  ↳ First In First Out

- #include <queue>

```
queue <string> q;
```

- q.size()
- q.push("Harsh");
  q.push("kunal");
  cout << q.front() << endl;
  q.pop();
  cout << q.front() << endl;

```
// screen
Harsh
kunal
```

→ Priority Queue

```
#include <iostream>
#include <queue>
using namespace std;
int main() {
  // max heap
    priority_queue<int> maxi;
  // min heap
priority_queue<int, vector<int>, greater<int>> mini;
maxi.push(4);
maxi.push(3);
maxi.push(2);
maxi.push(0);
cout << "size -> " << maxi.size() << endl;
  int n = maxi.size();
  for (int i=0; i<n; i++) {
    cout << maxi.top() << " ";
      maxi.pop();
    } cout << endl;
mini.push(5);
mini.push(1);
mini.push(0);
mini.push(4);
int m = mini.size();
 for (int i=0; i<m; i++) {
   cout << mini.top() << " ";
     mini.pop();
    } cout << endl;
cout << "Empty or not -> " << mini.empty() << endl;
}
```

O/P screen.

```
size -> 4
3 2 1 0
0 1 4 5
Empty or not -> 1
```

→ Set

- In set, only unique values are stored.
- Modification not possible, either insert or delete.
- Elements stored in sorted order.

- #include <set>

```
set<int> s;
s.insert(5); s.insert(5);
s.insert(3);
s.insert(6);
for(auto i:s){
cout << i<< endl; //356
}
```

↳ DO not store repeated values

- s.erase(s.begin()); //56
- s.count(5); //1
- set<int>::iterator itr = s.find(6);

check whether it is present or not

↳ tells the index.

- Time complexity of erase, find, insert, count → O(log n)
- Time complexity of size, begin, empty → O(1)

→ Map

```
#include <map>
map<int, string> m;
m[1] = "Dog";
m[2] = "Cat";
m[10] = "Horse";
m[3] = "Panda";
```

```
for auto (i:m){
  Cout << i.first << endl;
}
```

Similarly, i.second gives R.H.S. value.

o/p screen

1
2
3
10

Always in sorted order

```
m.erase(2);
Cout << "After erase" << endl;
for(auto i: m){
 Cout << i.first << " " << i.second << endl;
}
```

o/p screen

1 Dog
3 Panda
10 Horse

*. Algorithms

```
#include <algorithm>
#include <vector>
 vector <int> v;
  v.push_back (2);
  v.push_back (4);
  v.push_back (6);
  v.push_back (9);
Cout << binary_search (v.begin(), v.end(),5)
                << endl; //0 (i.e. false)
Cout << binary_search (v.begin(), v.end(), 6)
                << endl; //1 (i.e. True)
Cout << lower_bound (v.begin(), v.end(),6)
          - v.begin() << endl;
```

→ If returns the index //2

```
Cout << upper_bound (v.begin(),
v.end(), 6) - v.begin() << endl;
                //3
```

→
```
int a=3, b=5;
Cout << max(a,b); //5
Cout << min (a,b); //3
swap (a,b);
Cout << a << " " << b << endl;
```

// a → 5
b → 3

- String abcd = "abcd";
  reverse(abcd.begin(), abcd.end());
  Cout << "String →" << abcd << endl; //dcba